



DRAFT – CUPS Software Programmers Manual
CUPS-SPM-1.1

Easy Software Products
Copyright 1997–2000, All Rights Reserved

Table of Contents

<u>Preface</u>	1
<u>System Overview</u>	1
<u>Document Overview</u>	1
<u>1 – Printing System Overview</u>	3
<u>The Printing Problem</u>	3
<u>The Technology</u>	4
<u>Jobs</u>	4
<u>Classes</u>	4
<u>Filters</u>	4
<u>Printer Drivers</u>	5
<u>Networking</u>	5
<u>2 – The CUPS API</u>	7
<u>The CUPS Library</u>	7
<u>Detecting the CUPS Library in Autoconf</u>	7
<u>Basic Services</u>	7
<u>Include Files</u>	7
<u>Getting the Available Printers and Classes</u>	7
<u>Printing Files</u>	8
<u>Setting Printer Options</u>	8
<u>Cancelling Jobs</u>	8
<u>HTTP Services</u>	8
<u>Include Files</u>	8
<u>Connecting to a Server</u>	8
<u>Setting Request Fields</u>	8
<u>Issuing a Request</u>	8
<u>Getting the Request Status</u>	8
<u>Sending Request Data</u>	8
<u>Reading Request Data</u>	8
<u>IPP Services</u>	8
<u>Include Files</u>	9
<u>Creating an IPP Request</u>	9
<u>Adding Attributes</u>	9
<u>Sending an IPP Request</u>	9
<u>Reading an IPP Response</u>	9
<u>Finding Attributes</u>	9
<u>Looping Through Attributes</u>	9
<u>IPP Standard Operations</u>	9
<u>IPP Extension Operations</u>	9
<u>CUPS Extension Operations</u>	9
<u>Language Services</u>	9
<u>Include Files</u>	9
<u>Getting the Default Language</u>	10
<u>Getting the Language Encoding</u>	10
<u>Getting a Language String</u>	10
<u>PPD Services</u>	10

Table of Contents

Include Files	10
Loading a PPD File	10
Options and Groups	10
Finding an Option	10
Finding a Page Size	10
Marking Options	10
Checking for Conflicts	10
Sending Options	10
3 – Writing Filters	11
Overview	11
Security Considerations	11
Temporary Files	11
Page Accounting	11
Command-Line Arguments	11
Copy Generation	12
Environment Variables	12
Writing a HTML Filter	12
4 – Writing Printer Drivers	13
Overview	13
Page Accounting	13
Color Management	13
Raster Functions	13
cupsRasterOpen()	13
cupsRasterReadHeader()	14
cupsRasterReadPixels()	14
cupsRasterClose()	14
Writing a HP-PCL Driver	14
5 – Writing Backends	15
Overview	15
Security Considerations	15
Temporary Files	15
Page Accounting	15
Retries	15
Command-Line Arguments	16
Copy Generation	16
Environment Variables	16
Writing a Serial Port Backend	16
A – Constants	17
CUPS Constants	17
HTTP Constants	17
IPP Constants	17
Language Constants	17
PPD Constants	18

Table of Contents

<u>Raster Constants</u>	18
<u>B – Structures</u>	19
<u>C – Functions</u>	21
<u>cupsAddOption()</u>	22
<u>Usage</u>	22
<u>Arguments</u>	22
<u>Returns</u>	22
<u>Description</u>	22
<u>Example</u>	22
<u>See Also</u>	23
<u>cupsCancelJob()</u>	24
<u>Usage</u>	24
<u>Arguments</u>	24
<u>Returns</u>	24
<u>Description</u>	24
<u>Example</u>	24
<u>See Also</u>	24
<u>cupsDoFileRequest()</u>	25
<u>Usage</u>	25
<u>Arguments</u>	25
<u>Returns</u>	25
<u>Description</u>	25
<u>Example</u>	25
<u>See Also</u>	26
<u>cupsDoRequest()</u>	27
<u>Usage</u>	27
<u>Arguments</u>	27
<u>Returns</u>	27
<u>Description</u>	27
<u>Example</u>	27
<u>See Also</u>	28
<u>cupsFreeOptions()</u>	29
<u>Usage</u>	29
<u>Arguments</u>	29
<u>Description</u>	29
<u>Example</u>	29
<u>See Also</u>	29
<u>cupsGetClasses()</u>	30
<u>Usage</u>	30
<u>Arguments</u>	30
<u>Returns</u>	30
<u>Description</u>	30
<u>Example</u>	30
<u>See Also</u>	30
<u>cupsGetDefault()</u>	31

Table of Contents

<u>Usage</u>	31
<u>Returns</u>	31
<u>Description</u>	31
<u>Example</u>	31
<u>See Also</u>	31
<u>cupsGetOption()</u>	32
<u>Usage</u>	32
<u>Arguments</u>	32
<u>Returns</u>	32
<u>Description</u>	32
<u>See Also</u>	32
<u>cupsGetPassword()</u>	33
<u>Usage</u>	33
<u>Arguments</u>	33
<u>Returns</u>	33
<u>Description</u>	33
<u>Example</u>	33
<u>See Also</u>	33
<u>cupsGetPPD()</u>	34
<u>Usage</u>	34
<u>Arguments</u>	34
<u>Returns</u>	34
<u>Description</u>	34
<u>Example</u>	34
<u>cupsGetPrinters()</u>	35
<u>Usage</u>	35
<u>Arguments</u>	35
<u>Returns</u>	35
<u>Description</u>	35
<u>Example</u>	35
<u>See Also</u>	35
<u>cupsLangDefault()</u>	36
<u>Usage</u>	36
<u>Returns</u>	36
<u>Description</u>	36
<u>Example</u>	36
<u>See Also</u>	36
<u>cupsLangEncoding()</u>	37
<u>Usage</u>	37
<u>Arguments</u>	37
<u>Returns</u>	37
<u>Description</u>	37
<u>Example</u>	37
<u>See Also</u>	37
<u>cupsLangFlush()</u>	38
<u>Usage</u>	38
<u>Description</u>	38

Table of Contents

Example	38
See Also	38
cupsLangFree()	39
Usage	39
Arguments	39
Description	39
Example	39
See Also	39
cupsLangGet()	40
Usage	40
Arguments	40
Returns	40
Description	40
Example	40
See Also	40
cupsLangString()	41
Usage	41
Arguments	41
Returns	41
Description	41
Example	41
See Also	41
cupsLastError()	42
Usage	42
Returns	42
Description	42
Example	42
See Also	42
cupsMarkOptions()	43
Usage	43
Arguments	43
Returns	43
Description	43
Example	43
See Also	43
cupsParseOptions()	44
Usage	44
Arguments	44
Returns	44
Description	44
Example	44
See Also	44
cupsPrintFile()	45
Usage	45
Arguments	45
Returns	45
Description	45

Table of Contents

Example	45
See Also	46
cupsRasterClose()	47
Usage	47
Arguments	47
Description	47
Example	47
See Also	47
cupsRasterOpen()	48
Usage	48
Arguments	48
Returns	48
Description	48
Example	48
See Also	48
cupsRasterReadHeader()	49
Usage	49
Arguments	49
Returns	49
Description	49
Example	49
See Also	50
cupsRasterReadPixels()	51
Usage	51
Arguments	51
Returns	51
Description	51
Example	51
See Also	52
cupsRasterWriteHeader()	53
Usage	53
Arguments	53
Returns	53
Description	53
Example	53
See Also	53
cupsRasterWritePixels()	54
Usage	54
Arguments	54
Returns	54
Description	54
Example	54
See Also	55
cupsServer()	56
Usage	56
Returns	56
Description	56

Table of Contents

Example	56
See Also	56
cupsTempFile()	57
Usage	57
Arguments	57
Returns	57
Description	57
Example	57
cupsUser()	58
Usage	58
Returns	58
Description	58
Example	58
See Also	58
httpBlocking()	59
Usage	59
Arguments	59
Returns	59
Description	59
Example	59
See Also	59
httpCheck()	60
Usage	60
Arguments	60
Returns	60
Description	60
Example	60
See Also	60
httpClearFields()	61
Usage	61
Arguments	61
Returns	61
Description	61
Example	61
See Also	61
httpClose()	62
Usage	62
Arguments	62
Returns	62
Description	62
Example	62
See Also	62
httpConnect()	63
Usage	63
Arguments	63
Returns	63
Description	63

Table of Contents

Example	63
See Also	63
httpDecode64()	64
Usage	64
Arguments	64
Returns	64
Description	64
Example	64
See Also	64
httpDelete()	65
Usage	65
Arguments	65
Returns	65
Description	65
Example	65
See Also	65
httpEncode64()	66
Usage	66
Arguments	66
Returns	66
Description	66
Example	66
See Also	66
httpError()	67
Usage	67
Arguments	67
Returns	67
Description	67
Example	67
See Also	67
httpFlush()	68
Usage	68
Arguments	68
Returns	68
Description	68
Example	68
See Also	68
httpGet()	69
Usage	69
Arguments	69
Returns	69
Description	69
Example	69
See Also	69
httpGets()	70
Usage	70
Arguments	70

Table of Contents

Returns	70
Description	70
Example	70
See Also	70
httpGetString()	71
Usage	71
Arguments	71
Returns	71
Description	71
Example	71
See Also	71
httpGetDateTime()	72
Usage	72
Arguments	72
Returns	72
Description	72
Example	72
See Also	72
httpGetField()	73
Usage	73
Arguments	73
Returns	73
Description	73
Example	73
See Also	73
httpGetLength()	74
Usage	74
Arguments	74
Returns	74
Description	74
Example	74
See Also	74
httpHead()	75
Usage	75
Arguments	75
Returns	75
Description	75
Example	75
See Also	75
httpInitialize()	76
Usage	76
Arguments	76
Returns	76
Description	76
Example	76
See Also	76
httpOptions()	77

Table of Contents

Usage	77
Arguments	77
Returns	77
Description	77
Example	77
See Also	77
httpPost()	78
Usage	78
Arguments	78
Returns	78
Description	78
Example	78
See Also	78
httpPrintf()	79
Usage	79
Arguments	79
Returns	79
Description	79
Example	79
See Also	79
httpPut()	80
Usage	80
Arguments	80
Returns	80
Description	80
Example	80
See Also	80
httpRead()	81
Usage	81
Arguments	81
Returns	81
Description	81
Example	81
See Also	81
httpReconnect()	82
Usage	82
Arguments	82
Returns	82
Description	82
Example	82
See Also	82
httpSeparate()	83
Usage	83
Arguments	83
Returns	83
Description	83
Example	83

Table of Contents

See Also	83
httpSetField()	84
Usage	84
Arguments	84
Returns	84
Description	84
Example	84
See Also	84
httpTrace()	85
Usage	85
Arguments	85
Returns	85
Description	85
Example	85
See Also	85
httpUpdate()	86
Usage	86
Arguments	86
Returns	86
Description	86
Example	86
See Also	86
httpWrite()	87
Usage	87
Arguments	87
Returns	87
Description	87
Example	87
See Also	87
ippAddBoolean()	88
Usage	88
Arguments	88
Returns	88
Description	88
Example	88
See Also	88
ippAddBooleans()	89
Usage	89
Arguments	89
Returns	89
Description	89
Example	89
See Also	89
ippAddDate()	90
Usage	90
Arguments	90
Returns	90

Table of Contents

Description	90
Example	90
See Also	90
ippAddInteger()	91
Usage	91
Arguments	91
Returns	91
Description	91
Example	91
See Also	91
ippAddIntegers()	92
Usage	92
Arguments	92
Returns	92
Description	92
Example	92
See Also	92
ippAddRange()	93
Usage	93
Arguments	93
Returns	93
Description	93
Example	93
See Also	93
ippAddRanges()	94
Usage	94
Arguments	94
Returns	94
Description	94
Example	94
See Also	94
ippAddResolution()	95
Usage	95
Arguments	95
Returns	95
Description	95
Example	95
See Also	95
ippAddResolutions()	96
Usage	96
Arguments	96
Returns	96
Description	96
Example	96
See Also	96
ippAddSeparator()	97
Usage	97

Table of Contents

<u>Arguments</u>	97
<u>Returns</u>	97
<u>Description</u>	97
<u>Example</u>	97
<u>See Also</u>	97
<u>ippAddString()</u>	98
<u>Usage</u>	98
<u>Arguments</u>	98
<u>Returns</u>	98
<u>Description</u>	98
<u>Example</u>	98
<u>See Also</u>	98
<u>ippAddStrings()</u>	99
<u>Usage</u>	99
<u>Arguments</u>	99
<u>Returns</u>	99
<u>Description</u>	99
<u>Example</u>	99
<u>See Also</u>	99
<u>ippDateToTime()</u>	100
<u>Usage</u>	100
<u>Arguments</u>	100
<u>Returns</u>	100
<u>Description</u>	100
<u>Example</u>	100
<u>See Also</u>	100
<u>ippDelete()</u>	101
<u>Usage</u>	101
<u>Arguments</u>	101
<u>Returns</u>	101
<u>Description</u>	101
<u>Example</u>	101
<u>See Also</u>	101
<u>ippFindAttribute()</u>	102
<u>Usage</u>	102
<u>Arguments</u>	102
<u>Returns</u>	102
<u>Description</u>	102
<u>Example</u>	102
<u>See Also</u>	102
<u>ippLength()</u>	103
<u>Usage</u>	103
<u>Arguments</u>	103
<u>Returns</u>	103
<u>Description</u>	103
<u>Example</u>	103
<u>See Also</u>	103

Table of Contents

<u>ippNew()</u>	104
<u>Usage</u>	104
<u>Arguments</u>	104
<u>Returns</u>	104
<u>Description</u>	104
<u>Example</u>	104
<u>See Also</u>	104
<u>ippPort()</u>	105
<u>Usage</u>	105
<u>Arguments</u>	105
<u>Returns</u>	105
<u>Description</u>	105
<u>Example</u>	105
<u>See Also</u>	105
<u>ippRead()</u>	106
<u>Usage</u>	106
<u>Arguments</u>	106
<u>Returns</u>	106
<u>Description</u>	106
<u>Example</u>	106
<u>See Also</u>	106
<u>ippTimeToDate()</u>	107
<u>Usage</u>	107
<u>Arguments</u>	107
<u>Returns</u>	107
<u>Description</u>	107
<u>Example</u>	107
<u>See Also</u>	107
<u>ippWrite()</u>	108
<u>Usage</u>	108
<u>Arguments</u>	108
<u>Returns</u>	108
<u>Description</u>	108
<u>Example</u>	108
<u>See Also</u>	108
<u>ppdClose()</u>	109
<u>Usage</u>	109
<u>Arguments</u>	109
<u>Returns</u>	109
<u>Description</u>	109
<u>Example</u>	109
<u>See Also</u>	109
<u>ppdConflicts()</u>	110
<u>Usage</u>	110
<u>Arguments</u>	110
<u>Returns</u>	110
<u>Description</u>	110

Table of Contents

Example	110
See Also	110
ppdEmitFd()	111
Usage	111
Arguments	111
Returns	111
Description	111
Example	111
See Also	111
ppdEmit()	112
Usage	112
Arguments	112
Returns	112
Description	112
Example	112
See Also	112
ppdFindChoice()	113
Usage	113
Arguments	113
Returns	113
Description	113
Example	113
See Also	113
ppdFindMarkedChoice()	114
Usage	114
Arguments	114
Returns	114
Description	114
Example	114
See Also	114
ppdFindOption()	115
Usage	115
Arguments	115
Returns	115
Description	115
Example	115
See Also	115
ppdIsMarked()	116
Usage	116
Arguments	116
Returns	116
Description	116
Example	116
See Also	116
ppdMarkDefaults()	117
Usage	117
Arguments	117

Table of Contents

Returns	117
Description	117
Example	117
See Also	117
ppdMarkOption()	118
Usage	118
Arguments	118
Returns	118
Description	118
Example	118
See Also	118
ppdOpenFd()	119
Usage	119
Arguments	119
Returns	119
Description	119
Example	119
See Also	119
ppdOpenFile()	120
Usage	120
Arguments	120
Returns	120
Description	120
Example	120
See Also	120
ppdOpen()	121
Usage	121
Arguments	121
Returns	121
Description	121
Example	121
See Also	121
ppdPageLength()	122
Usage	122
Arguments	122
Returns	122
Description	122
Example	122
See Also	122
ppdPageSize()	123
Usage	123
Arguments	123
Returns	123
Description	123
Example	123
See Also	123
ppdPageWidth()	124

Table of Contents

[Usage](#).....124

[Arguments](#).....124

[Returns](#).....124

[Description](#).....124

[Example](#).....124

[See Also](#).....124

Preface

This software programmers manual provides software programming information for the Common UNIX Printing System ("CUPS") Version 1.1.

System Overview

The Common UNIX Printing System provides a portable printing layer for UNIX® operating systems. It has been developed by [Easy Software Products](#) to promote a standard printing solution for all UNIX vendors and users. CUPS provides the System V and Berkeley command-line interfaces.

CUPS uses the Internet Printing Protocol (IETF-IPP) as the basis for managing print jobs and queues. The Line Printer Daemon (LPD, RFC1179), Server Message Block (SMB), and AppSocket protocols are also supported with reduced functionality.

CUPS adds network printer browsing and PostScript Printer Description ("PPD")-based printing options to support real world applications under UNIX.

CUPS also includes a customized version of GNU GhostScript (currently based off GNU GhostScript 5.50) and an image file RIP that is used to support non-PostScript printers.

Document Overview

This software administrators manual is organized into the following sections:

- 1 – Printing System Overview

- 2 – The CUPS API
- 3 – Writing Filters
- 4 – Writing Printer Drivers
- 5 – Writing Backends
- A – Constants
- B – Structures
- C – Functions

1 – Printing System Overview

This chapter provides an overview of how the Common UNIX Printing System works.

The Printing Problem

For years *the printing problem* has plagued UNIX®. Unlike Microsoft® Windows® or MacOS, UNIX has no standard interface or system in place for supporting printers. Among the solutions previously available, the Berkeley and System V printing systems are the most prevalent.

These printing systems support line printers (text only) or PostScript printers (text and graphics), and with some coaxing they can be made to support a full range of printers and file formats. However, because each variant of the UNIX operating system uses a different printing system than the next, developing printer drivers for a wide range of printers is extremely difficult. That combined with the limited volume of customers for each UNIX variant has forced most printer vendors to give up supporting UNIX entirely.

The Common UNIX Printing System, or CUPS, is designed to eliminate *the printing problem*. One common printing system can be used by all UNIX variants to support the printing needs of users. Printer vendors can use its modular filter interface to develop a single driver program that supports a wide range of file formats with little or no effort. Since CUPS provides both the System V and Berkeley printing commands, users (and applications) can reap the benefits of this new technology with no changes.

The Technology

CUPS is based upon an emerging Internet standard called the Internet Printing Protocol, or IPP. IPP has been embraced by dozens of printer and printer server manufacturers, and will be supported by the next Microsoft Windows operating system.

IPP defines a standard protocol for printing as well as managing print jobs and printer options like media size, resolution, and so forth. Like all IP-based protocols, IPP can be used locally or over the Internet to printers hundreds or thousands of miles away. Unlike other protocols, however, IPP also supports access control, authentication, and encryption, making it a much more secure printing solution than older ones.

IPP is layered on top of the Hyper-Text Transport Protocol, or HTTP, which is the basis of web servers on the Internet. This allows the user to view documentation and status information on a printer or server using their web browser.

CUPS provides a complete IPP/1.0-based printing system that provides Basic authentication and domain or IP-based access control. Digest authentication and TLS encryption will be available in future versions of CUPS.

Jobs

Each file that is submitted for printing is called a *job*. Jobs are identified by a unique number starting at 1 and are assigned to a particular destination (usually a printer). Jobs can also have options associated with them such as media size, number of copies, and priority.

Classes

CUPS supports collections of printers known as *classes*. Jobs sent to a class are forwarded to the first available printer in the class.

Filters

Filters allow a user or application to print many types of files without extra effort. Print jobs sent to a CUPS server are filtered before sending them to a printer. Some filters convert job files to different formats that the printer can understand. Others perform page selection and ordering tasks. *Backend* filters perform the most important task of all – they send the filtered print data to the printer.

CUPS provides filters for printing many types of image files, HP-GL/2 files, PDF files, and text files. CUPS also supplies PostScript and image file Raster Image Processors, or RIPs, that convert PostScript or image files into bitmaps that can be sent to a raster printer.

CUPS provides backends for printing over parallel and serial ports, and over the network via the JetDirect (AppSocket), Server Message Block, and Line Printer Daemon protocols.

Printer Drivers

Printer drivers in CUPS consist of one or more filters specific to a printer. CUPS includes a sample printer driver for Hewlett–Packard LaserJet and DeskJet printers. While this driver does not generate optimal output for different printer models, it does demonstrate how you can write your own printer drivers and incorporate them into CUPS.

Networking

Printers and classes on the local system are automatically shared with other systems on the network. This allows you to setup one system to print to a printer and use this system as a printer server or spool host for all of the others. If there is only one occurrence of a printer on a network, then that printer can be accessed using its name alone. If more than one printer exists with the same name, users must select the printer by specifying which server to use (e.g. "printer@host1" or "printer@host2".)

CUPS also provides *implicit classes*, which are collections of printers and/or classes with the same name. This allows you to setup multiple servers pointing to the same physical network printer, for example, so that you aren't relying on a single system for printing. Because this also works with printer classes, you can setup multiple servers and printers and never worry about a "single point of failure" unless all of the printers and servers goes down!

2 – The CUPS API

This chapter describes the CUPS Application Programmers Interface ("API").

The CUPS Library

Detecting the CUPS Library in Autoconf

Basic Services

Include Files

Getting the Available Printers and Classes

Printing Files

Setting Printer Options

Cancelling Jobs

HTTP Services

Include Files

Connecting to a Server

Setting Request Fields

Issuing a Request

Getting the Request Status

Sending Request Data

Reading Request Data

IPP Services

Include Files

Creating an IPP Request

Adding Attributes

Sending an IPP Request

Reading an IPP Response

Finding Attributes

Looping Through Attributes

IPP Standard Operations

IPP Extension Operations

CUPS Extension Operations

Language Services

Include Files

Getting the Default Language

Getting the Language Encoding

Getting a Language String

PPD Services

Include Files

Loading a PPD File

Options and Groups

Finding an Option

Finding a Page Size

Marking Options

Checking for Conflicts

Sending Options

3 – Writing Filters

This chapter describes how to write a file filter for CUPS.

Overview

Security Considerations

Users and Groups

Temporary Files

Page Accounting

Command-Line Arguments

Copy Generation

Environment Variables

Writing a HTML Filter

4 – Writing Printer Drivers

This chapter discusses how to write a printer driver, which is a special filter program that converts CUPS raster data into the appropriate commands and data required for a printer.

Overview

Page Accounting

Color Management

Raster Functions

cupsRasterOpen()

cupsRasterReadHeader()

cupsRasterReadPixels()

cupsRasterClose()

Writing a HP-PCL Driver

5 – Writing Backends

This chapter describes how to write a backend for CUPS. Backends communicate directly with printers and allow printer drivers and filters to send data using any type of connection transparently.

Overview

Security Considerations

Users and Groups

Temporary Files

Page Accounting

Retries

Command-Line Arguments

Copy Generation

Environment Variables

Writing a Serial Port Backend

A – Constants

This appendix lists all of the constants that are defined by the CUPS API.

CUPS Constants

HTTP Constants

IPP Constants

Language Constants

PPD Constants

Raster Constants

B – Structures

This appendix describes all of the structures that are defined by the CUPS API.

C – Functions

This appendix provides a reference for all of the CUPS API functions.

cupsAddOption()

Usage

```
int
cupsAddOption(const char *name,
              const char *value,
              int num_options,
              cups_option_t **options);
```

Arguments

Argument	Description
name	The name of the option.
value	The value of the option.
num_options	Number of options currently in the array.
options	Pointer to the options array.

Returns

The new number of options.

Description

cupsAddOption() adds an option to the specified array.

Example

```
#include <cups.h>

...

/* Declare the options array */
int      num_options;
cups_option_t *options;

/* Initialize the options array */
num_options = 0;
options      = (cups_option_t *)0;

/* Add options using cupsAddOption() */
num_options = cupsAddOption("media", "letter", num_options, &options);
num_options = cupsAddOption("resolution", "300dpi", num_options, &options);
```

See Also

[cupsFreeOptions\(\)](#), [cupsGetOption\(\)](#), [cupsParseOptions\(\)](#)

cupsCancelJob()

Usage

```
int  
cupsCancelJob(const char *dest,  
              int job);
```

Arguments

Argument	Description
dest	Printer or class name
job	Job ID

Returns

1 on success, 0 on failure. On failure the error can be found by calling [cupsLastError\(\)](#).

Description

`cupsCancelJob()` cancels the specifies job.

Example

```
#include <cups.h>  
  
cupsCancelJob("LaserJet", 1);
```

See Also

[cupsLastError\(\)](#), [cupsPrintFile\(\)](#)

cupsDoFileRequest()

Usage

```
ipp_t *
cupsDoFileRequest(http_t *http,
                  ipp_t *request,
                  const char *resource,
                  const char *filename);
```

Arguments

Argument	Description
http	HTTP connection to server.
request	IPP request data.
resource	HTTP resource name for POST.
filename	File to send with POST request (NULL pointer if none.)

Returns

IPP response data or NULL if the request fails. On failure the error can be found by calling [cupsLastError\(\)](#).

Description

`cupsDoFileRequest()` does a HTTP POST request and provides the IPP request and optionally the contents of a file to the IPP server. It also handles resubmitting the request and performing password authentication as needed.

Example

```
#include <cups.h>

http_t      *http;
cups_lang_t *language;
ipp_t       *request;
ipp_t       *response;

...

/* Get the default language */
language = cupsLangDefault();

/* Create a new IPP request */
request = ippNew();

request->request.op.operation_id = IPP_PRINT_FILE;

cupsDoFileRequest()
```

```
request->request.op.request_id    = 1;

/* Add required attributes */
ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_CHARSET,
    "attributes-charset", NULL, cupsLangEncoding(language));

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_LANGUAGE,
    "attributes-natural-language", NULL,
    language != NULL ? language->language : "C");

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_URI, "printer-uri",
    NULL, "ipp://hostname/resource");

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_NAME, "requesting-user-name",
    NULL, cupsUser());

/* Do the request... */
response = cupsDoFileRequest(http, request, "/resource", "filename.txt");
```

See Also

[cupsLangDefault\(\)](#), [cupsLangEncoding\(\)](#), [cupsUser\(\)](#), [httpConnect\(\)](#),
[ippAddString\(\)](#), [ippNew\(\)](#)

cupsDoRequest()

Usage

```
ipp_t *
cupsDoRequest(http_t *http,
              ipp_t *request,
              const char *resource);
```

Arguments

Argument	Description
http	HTTP connection to server.
request	IPP request data.
resource	HTTP resource name for POST.

Returns

IPP response data or NULL if the request fails. On failure the error can be found by calling [cupsLastError\(\)](#).

Description

`cupsDoRequest()` does a HTTP POST request and provides the IPP request to the IPP server. It also handles resubmitting the request and performing password authentication as needed.

Example

```
#include <cups.h>

http_t      *http;
cups_lang_t *language;
ipp_t       *request;
ipp_t       *response;

...

/* Get the default language */
language = cupsLangDefault();

/* Create a new IPP request */
request = ippNew();

request->request.op.operation_id = IPP_GET_PRINTER_ATTRIBUTES;
request->request.op.request_id   = 1;

/* Add required attributes */
ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_CHARSET,
```

`cupsDoRequest()`

```
"attributes-charset", NULL, cupsLangEncoding(language));

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_LANGUAGE,
             "attributes-natural-language", NULL,
             language != NULL ? language->language : "C");

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_URI, "printer-uri",
             NULL, "ipp://hostname/resource");

/* Do the request... */
response = cupsDoRequest(http, request, "/resource");
```

See Also

[cupsLangDefault\(\)](#), [cupsLangEncoding\(\)](#), [cupsUser\(\)](#), [httpConnect\(\)](#),
[ippAddString\(\)](#), [ippNew\(\)](#)

cupsFreeOptions()

Usage

```
void  
cupsFreeOptions(int num_options,  
                cups_option_t *options);
```

Arguments

Argument	Description
num_options	Number of options in array.
options	Pointer to options array.

Description

`cupsFreeOptions()` frees all memory associated with the option array specified.

Example

```
#include <cups/cups.h>  
  
int          num_options;  
cups_option_t *options;  
  
...  
  
cupsFreeOptions(num_options, options);
```

See Also

[cupsAddOption\(\)](#), [cupsGetOption\(\)](#), [cupsMarkOptions\(\)](#), [cupsParseOptions\(\)](#)

cupsGetClasses()

Usage

```
int
cupsGetClasses(char ***classes);
```

Arguments

Argument	Description
classes	Pointer to character pointer array.

Returns

The number of printer classes available.

Description

`cupsGetClasses()` gets a list of the available printer classes. The returned array should be freed using the `free()` when it is no longer needed.

Example

```
#include <cups/cups.h>

int i;
int num_classes;
char **classes;

...

num_classes = cupsGetClasses();

...

if (num_classes > 0)
{
    for (i = 0; i < num_classes; i++)
        free(classes[i]);

    free(classes);
}
```

See Also

[cupsGetDefault\(\)](#), [cupsGetPrinters\(\)](#)

cupsGetDefault()

Usage

```
const char *  
cupsGetDefault(void);
```

Returns

A pointer to the default destination.

Description

`cupsGetDefault()` gets the default destination printer or class. The default destination is stored in a static string and will be overwritten (usually with the same value) after each call.

Example

```
#include <cups/cups.h>  
  
printf("The default destination is %s\n", cupsGetDefault());
```

See Also

[`cupsGetClasses\(\)`, `cupsGetPrinters\(\)`](#)

cupsGetOption()

Usage

```
const char *
cupsGetOption(const char *name,
              int num_options,
              cups_option_t *options);
```

Arguments

Argument	Description
name	The name of the option.
num_options	The number of options in the array.
options	The options array.

Returns

A pointer to the option values or NULL if the option is not defined.

Description

`cupsGetOption()` returns the first occurrence of the named option. If the option is not included in the options array then a NULL pointer is returned.

```
#include <cups/cups.h>

int          num_options;
cups_option_t *options;
const char   *media;

...

media = cupsGetOption("media", num_options, options);
```

See Also

[cupsAddOption\(\)](#), [cupsFreeOptions\(\)](#), [cupsMarkOptions\(\)](#), [cupsParseOptions\(\)](#)

cupsGetPassword()

Usage

```
const char *  
cupsGetPassword(const char *prompt);
```

Arguments

Argument	Description
prompt	The prompt to display to the user.

Returns

A pointer to the password that was entered or NULL if no password was entered.

Description

`cupsGetPassword()` displays the prompt string and asks the user for a password. The password text is not echoed to the user.

Example

```
#include <cups/cups.h>  
  
char *password;  
  
...  
  
password = cupsGetPassword("Please enter a password:");
```

See Also

[cupsServer\(\)](#), [cupsUser\(\)](#)

cupsGetPPD()

Usage

```
const char *
cupsGetPPD(const char *printer);
```

Arguments

Argument	Description
printer	The name of the printer.

Returns

The name of a temporary file containing the PPD file or NULL if the printer cannot be located or does not have a PPD file.

Description

`cupsGetPPD()` gets a copy of the PPD file for the named printer. The printer name can be of the form "printer" or "printer@hostname".

You should remove (unlink) the PPD file after you are done using it. The filename is stored in a static buffer and will be overwritten with each call to `cupsGetPPD()`.

Example

```
#include <cups/cups.h>

char *ppd;

...

ppd = cupsGetPPD("printer@hostname");

...

unlink(ppd);
```

cupsGetPrinters()

Usage

```
int
cupsGetPrinters(char ***printers);
```

Arguments

Argument	Description
printers	Pointer to character pointer array.

Returns

The number of printer printers available.

Description

`cupsGetPrinters()` gets a list of the available printers. The returned array should be freed using the `free()` when it is no longer needed.

Example

```
#include <cups/cups.h>

int i;
int num_printers;
char **printers;

...

num_printers = cupsGetPrinters();

...

if (num_printers > 0)
{
    for (i = 0; i < num_printers; i++)
        free(printers[i]);

    free(printers);
}
```

See Also

[cupsGetClasses\(\)](#), [cupsGetDefault\(\)](#)

cupsLangDefault()

Usage

```
const char *  
cupsLangDefault(void);
```

Returns

A pointer to the default language structure.

Description

`cupsLangDefault()` returns a language structure for the default language. The default language is defined by the `LANG` environment variable. If the specified language cannot be located then the POSIX (English) locale is used.

Call `cupsLangFree()` to free any memory associated with the language structure when you are done.

Example

```
#include <cups/language.h>  
  
cups_lang_t *language;  
...  
  
language = cupsLangDefault();  
  
...  
  
cupsLangFree(language);
```

See Also

[cupsLangEncoding\(\)](#), [cupsLangFlush\(\)](#), [cupsLangFree\(\)](#), [cupsLangGet\(\)](#), [cupsLangString\(\)](#)

cupsLangEncoding()

Usage

```
char *
cupsLangEncoding(cups_lang_t *language);
```

Arguments

Argument	Description
language	The language structure.

Returns

A pointer to the encoding string.

Description

`cupsLangEncoding()` returns the language encoding used for the specified language, e.g. "iso-8859-1", "utf-8", etc.

Example

```
#include <cups/language.h>

cups_lang_t *language;
char        *encoding;
...

language = cupsLangDefault();
encoding = cupsLangEncoding(language);
...

cupsLangFree(language);
```

See Also

[cupsLangDefault\(\)](#), [cupsLangFlush\(\)](#), [cupsLangFree\(\)](#), [cupsLangGet\(\)](#), [cupsLangString\(\)](#)

cupsLangFlush()

Usage

```
void  
cupsLangFlush(void);
```

Description

`cupsLangFlush()` frees all language structures that have been allocated.

Example

```
#include <cups/language.h>  
  
...  
  
cupsLangFlush();
```

See Also

[cupsLangDefault\(\)](#), [cupsLangEncoding\(\)](#), [cupsLangFree\(\)](#), [cupsLangGet\(\)](#), [cupsLangString\(\)](#)

cupsLangFree()

Usage

```
void
cupsLangFree(cups_lang_t *language);
```

Arguments

Argument	Description
language	The language structure to free.

Description

cupsLangFree() frees the specified language structure.

Example

```
#include <cups/language.h>

cups_lang_t *language;
...

cupsLangFree(language);
```

See Also

[cupsLangDefault\(\)](#), [cupsLangEncoding\(\)](#), [cupsLangFlush\(\)](#), [cupsLangGet\(\)](#), [cupsLangString\(\)](#)

cupsLangGet()

Usage

```
cups_lang_t *
cupsLangGet(const char *name);
```

Arguments

Argument	Description
name	The name of the locale.

Returns

A pointer to a language structure.

Description

`cupsLangGet()` returns a language structure for the specified locale. If the locale is not defined then the POSIX (English) locale is substituted.

Example

```
#include <cups/language.h>

cups_lang_t *language;

...

language = cupsLangGet("fr");

...

cupsLangFree(language);
```

See Also

[cupsLangDefault\(\)](#), [cupsLangEncoding\(\)](#), [cupsLangFlush\(\)](#), [cupsLangFree\(\)](#), [cupsLangString\(\)](#)

cupsLangString()

Usage

```
char *
cupsLangString(cups_lang_t *language,
               int          message);
```

Arguments

Argument	Description
language	The language to query.
message	The message number.

Returns

A pointer to the message string or NULL if the message is not defined.

Description

`cupsLangString()` returns a pointer to the specified message string in the specified language.

Example

```
#include <cups/language.h>

cups_lang_t *language;
char          *s;
...

language = cupsLangGet("fr");

s = cupsLangString(language, CUPS_MSG_YES);

...

cupsLangFree(language);
```

See Also

[cupsLangDefault\(\)](#), [cupsLangEncoding\(\)](#), [cupsLangFlush\(\)](#), [cupsLangFree\(\)](#), [cupsLangGet\(\)](#)

cupsLastError()

Usage

```
ipp_status_t  
cupsLastError(void);
```

Returns

An enumeration containing the last IPP error.

Description

`cupsLastError()` returns the last IPP error that occurred. If no error occurred then it will return `IPP_OK` or `IPP_OK_CONFLICT`.

Example

```
#include <cups/cups.h>  
  
ipp_status_t status;  
  
...  
  
status = cupsLastError();
```

See Also

[cupsCancelJob\(\)](#), [cupsPrintFile\(\)](#)

cupsMarkOptions()

Usage

```
int
cupsMarkOptions(ppd_file_t *ppd,
                int num_options,
                cups_option_t *options);
```

Arguments

Argument	Description
ppd	The PPD file to mark.
num_options	The number of options in the options array.
options	A pointer to the options array.

Returns

The number of conflicts found.

Description

`cupsMarkOptions()` marks options in the PPD file. It also handles mapping of IPP option names and values to PPD option names.

Example

```
#include <cups/cups.h>

int          num_options;
cups_option_t *options;
ppd_file_t   *ppd;

...

cupsMarkOptions(ppd, num_options, options);
```

See Also

[cupsAddOption\(\)](#), [cupsFreeOptions\(\)](#), [cupsGetOption\(\)](#), [cupsParseOptions\(\)](#)

cupsParseOptions()

Usage

```
int
cupsParseOptions(const char *arg,
                 int num_options,
                 cups_option_t **options);
```

Arguments

Argument	Description
arg	The string containing one or more options.
num_options	The number of options in the options array.
options	A pointer to the options array pointer.

Returns

The new number of options in the array.

Description

`cupsParseOptions()` parses the specifies string for one or more options of the form "name=value", "name", or "noname". It can be called multiple times to combine the options from several strings.

Example

```
#include <cups/cups.h>

int          num_options;
cups_option_t *options;

...

num_options = 0;
options     = (cups_option_t *)0;
num_options = cupsParseOptions(argv[5], num_options, &options);
```

See Also

[cupsAddOption\(\)](#), [cupsFreeOptions\(\)](#), [cupsGetOption\(\)](#), [cupsMarkOptions\(\)](#)

cupsPrintFile()

Usage

```
int
cupsPrintFile(const char *printer,
              const char *filename,
              const char *title,
              int num_options,
              cups_option_t *options);
```

Arguments

Argument	Description
printer	The printer or class to print to.
filename	The file to print.
title	The job title.
num_options	The number of options in the options array.
options	A pointer to the options array.

Returns

The new job ID number or 0 on error.

Description

`cupsPrintFile()` sends a file to the specified printer or class for printing. If the job cannot be printed the error code can be found by calling `cupsLastError()`.

Example

```
#include <cups/cups.h>

int          num_options;
cups_option_t *options;

...

cupsPrintFile("printer@hostname", "filename.ps", "Job Title", num_options,
              options);
```

See Also

[cupsCancelJob\(\)](#), [cupsLastError\(\)](#)

cupsRasterClose()

Usage

```
void  
cupsRasterClose(cups_raster_t *ras);
```

Arguments

Argument	Description
ras	The raster stream to close.

Description

`cupsRasterClose()` closes the specified raster stream.

Example

```
#include <cups/raster.h>  
  
cups_raster_t *ras;  
  
...  
  
cupsRasterClose(ras);
```

See Also

[cupsRasterOpen\(\)](#), [cupsRasterReadHeader\(\)](#), [cupsRasterReadPixels\(\)](#), [cupsRasterWriteHeader\(\)](#), [cupsRasterWritePixels\(\)](#)

cupsRasterOpen()

Usage

```
cups_raster_t *
cupsRasterOpen(int fd,
               cups_mode_t mode);
```

Arguments

Argument	Description
fd	The file descriptor to use.
mode	The mode to use; CUPS_RASTER_READ or CUPS_RASTER_WRITE.

Returns

A pointer to a raster stream or NULL if there was an error.

Description

`cupsRasterOpen()` opens a raster stream for reading or writing.

Example

```
#include <cups/raster.h>

cups_raster_t *ras;

...

ras = cupsRasterOpen(0, CUPS_RASTER_READ);
```

See Also

[cupsRasterClose\(\)](#), [cupsRasterReadHeader\(\)](#), [cupsRasterReadPixels\(\)](#), [cupsRasterWriteHeader\(\)](#), [cupsRasterWritePixels\(\)](#)

cupsRasterReadHeader()

Usage

```
unsigned
cupsRasterReadHeader(cups_raster_t *ras,
                    cups_page_header_t *header);
```

Arguments

Argument	Description
ras	The raster stream to read from.
header	A pointer to a page header structure to read into.

Returns

1 on success, 0 on EOF or error.

Description

`cupsRasterReadHeader()` reads a page header from the specified raster stream.

Example

```
#include <cups/raster.h>

int          line;
cups_raster_t *ras;
cups_raster_header_t header;
unsigned char pixels[8192];
...

while (cupsRasterReadHeader(ras, &header))
{
    ...

    for (line = 0; line < header.cupsHeight; line++)
    {
        cupsRasterReadPixels(ras, pixels, header.cupsBytesPerLine);

        ...
    }
}
```

See Also

[cupsRasterClose\(\)](#), [cupsRasterOpen\(\)](#), [cupsRasterReadPixels\(\)](#), [cupsRasterWriteHeader\(\)](#),
[cupsRasterWritePixels\(\)](#)

cupsRasterReadPixels()

Usage

```
unsigned
cupsRasterReadPixels(cups_raster_t *ras,
                    unsigned char *pixels,
                    unsigned length);
```

Arguments

Argument	Description
ras	The raster stream to read from.
pixels	The pointer to a pixel buffer.
length	The number of bytes of pixel data to read.

Returns

The number of bytes read or 0 on EOF or error.

Description

`cupsRasterReadPixels()` reads pixel data from the specified raster stream.

Example

```
#include <cups/raster.h>

int          line;
cups_raster_t *ras;
cups_raster_header_t header;
unsigned char pixels[8192];
...

while (cupsRasterReadHeader(ras, &header))
{
    ...

    for (line = 0; line < header.cupsHeight; line++)
    {
        cupsRasterReadPixels(ras, pixels, header.cupsBytesPerLine);

        ...
    }
}
```

See Also

[cupsRasterClose\(\)](#), [cupsRasterOpen\(\)](#), [cupsRasterReadHeader\(\)](#), [cupsRasterWriteHeader\(\)](#),
[cupsRasterWritePixels\(\)](#)

cupsRasterWriteHeader()

Usage

```
unsigned
cupsRasterWriteHeader(cups_raster_t *ras,
                     cups_page_header_t *header);
```

Arguments

Argument	Description
ras	The raster stream to write to.
header	A pointer to the page header to write.

Returns

1 on success, 0 on error.

Description

`cupsRasterWriteHeader()` writes the specified page header to a raster stream.

Example

```
#include <cups/raster.h>

int          line;
cups_raster_t *ras;
cups_raster_header_t header;
unsigned char pixels[8192];
...

cupsRasterWriteHeader(ras, &header);

for (line = 0; line < header.cupsHeight; line++)
{
    ...

    cupsRasterWritePixels(ras, pixels, header.cupsBytesPerLine);
}
```

See Also

[cupsRasterClose\(\)](#), [cupsRasterOpen\(\)](#), [cupsRasterReadHeader\(\)](#), [cupsRasterReadPixels\(\)](#), [cupsRasterWritePixels\(\)](#)

cupsRasterWritePixels()

Usage

```
unsigned
cupsRasterWritePixels(cups_raster_t *ras,
                     unsigned char *pixels,
                     unsigned length);
```

Arguments

Argument	Description
ras	The raster stream to write to.
pixels	The pixel data to write.
length	The number of bytes to write.

Returns

The number of bytes written.

Description

`cupsRasterWritePixels()` writes the specified pixel data to a raster stream.

Example

```
#include <cups/raster.h>

int          line;
cups_raster_t *ras;
cups_raster_header_t header;
unsigned char pixels[8192];
...

cupsRasterWriteHeader(ras, &header);

for (line = 0; line < header.cupsHeight; line++)
{
    ...

    cupsRasterWritePixels(ras, pixels, header.cupsBytesPerLine);
}
```

See Also

[cupsRasterClose\(\)](#), [cupsRasterOpen\(\)](#), [cupsRasterReadHeader\(\)](#), [cupsRasterReadPixels\(\)](#),
[cupsRasterWriteHeader\(\)](#)

cupsServer()

Usage

```
const char *  
cupsServer(void);
```

Returns

A pointer to the default server name.

Description

`cupsServer()` returns a pointer to the default server name. The server name is stored in a static location and will be overwritten with every call to `cupsServer()`

The default server is determined from the following locations:

1. The `CUPS_SERVER` environment variable,
2. The `ServerName` directive in the *cupsd.conf* file,
3. The default host, "localhost".

Example

```
#include <cups/cups.h>  
  
const char *server;  
  
server = cupsServer();
```

See Also

[cupsGetPassword\(\)](#), [cupsUser\(\)](#)

cupsTempFile()

Usage

```
char *
cupsTempFile(char *filename,
             int length);
```

Arguments

Argument	Description
filename	The character string to hold the temporary filename.
length	The size of the filename string in bytes.

Returns

A pointer to filename.

Description

`cupsTempFile()` generates a temporary filename for the */var/tmp* directory or the directory specified by the `TMPDIR` environment variable.

Example

```
#include <cups/cups.h>

char filename[256];

cupsTempFile(filename, sizeof(filename));
```

cupsUser()

Usage

```
const char *  
cupsUser(void);
```

Returns

A pointer to the current username or NULL if the user ID is undefined.

Description

`cupsUser()` returns the name associated with the current user ID as reported by the `getuid()` system call.

Example

```
#include <cups/cups.h>  
  
const char *user;  
  
user = cupsUser();
```

See Also

[cupsGetPassword\(\)](#), [cupsServer\(\)](#)

httpBlocking()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpCheck()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpClearFields()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpClose()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpConnect()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpDecode64()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpDelete()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpEncode64()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpError()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpFlush()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGet()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGets()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetString()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetDateTime()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetField()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpGetLength()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpHead()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpInitialize()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpOptions()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpPost()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpPrintf()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpPut()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpRead()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpReconnect()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpSeparate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpSetField()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpTrace()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpUpdate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

httpWrite()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddBoolean()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddBooleans()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddDate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddInteger()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddIntegers()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddRange()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddRanges()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddResolution()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddResolutions()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddSeparator()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddString()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippAddStrings()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippDateToTime()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippDelete()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippFindAttribute()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippLength()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippNew()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippPort()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippRead()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippTimeToDate()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ippWrite()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdClose()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdConflicts()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

pddEmitFd()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdEmit()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdFindChoice()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdFindMarkedChoice()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdFindOption()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdIsMarked()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdMarkDefaults()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdMarkOption()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdOpenFd()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdOpenFile()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdOpen()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdPageLength()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdPageSize()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also

ppdPageWidth()

Usage

Arguments

Argument	Description

Returns

Description

Example

See Also